[19]中华人民共和国国家知识产权局

[51]Int.Cl[6]
G06K 9/00

[12]发明专利申请公开说明书

[21]申请号 98125537.X

[43]公开日 1999年9月8日

[11]公开号 CN 1227940A

[22]申请日 98.12.21 [21]申请号 98125537.X
[30]优先权
[32]97.12.22 [33]US [31]08/995629
[71]申请人 国际商业机器公司
地址 美国纽约州
[72]发明人 J·L·泰勒

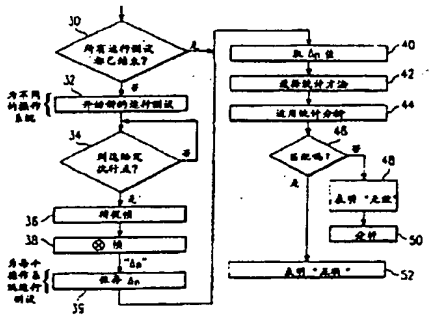
[74]专利代理机构 中国专利代理(香港)有限公司
代理人 邹光新 王忠忠

权利要求书4页 说明书9页 附图页数2页

[54]发明名称 用于确认应用程序图形显示输出的方法和系统

[57]摘要

一种方法，用于确认一个给定软件应用程序的图形显示输出在各操作系统上是一致的。首先，在每个操作系统上，在该软件应用程序的给定执行点处，捕捉连续的图形显示输出。然后，处理该对帧，以生成一个△值，该值为各操作系统屏蔽掉在该对帧上非时变的图形显示输出结构。然后，比较在每次运行测试中生成的△值(最好使用一个统计分析)，以确认该给定应用程序的图形显示输出在各操作系统上是一致的，本技术也可被用于在多次执行(该多次执行运行在一个同类计算机系统中)中确认该应用程序。



ISSN1008-4274

权　利　要　求　书

1. 一种方法，用于确认一个给定软件应用程序的图形显示输出在各个操作系统上是一致的，该方法的特点在于，它包括以下步骤:

在每个操作系统上，在该软件应用程序的一个给定执行点处，捕捉一对图形显示输出帧;

为各操作系统处理每对图形输出帧，以生成给定信息，该给定信息为各操作系统屏蔽掉在一对帧中非时变的图形显示输出结构;

比较为各个操作系统所生成的给定信息，以确认在各操作系统上，该给定软件应用程序的图形显示输出满足一个给定准则.

2. 如权利要求 1 所述的方法，其特征在于，该给定的软件应用程序是一个用 Java 编写的应用程序.

3. 如权利要求 1 所述的方法，其特征在于该对帧是连续的图形显示输出帧.

4. 如权利要求 1 所述的方法，其特征在于，该比较步骤包括执行一个统计算法.

5. 如权利要求 1 所述的方法，其特征在于，该给定法则检验该给定信息在一个给定置信度内是否可比.

6. 如权利要求 5 所述的方法，其特征在于该给定置信度是用户可选的.

7. 如权利要求 1 所述的方法，其特征在于，该处理步骤包括对构成一对的帧执行异或逻辑操作.

8. 如权利要求 7 所述的方法，其特征在于，该异或逻辑操作是按位执行的.

9. 一种方法，用来确认在各个操作系统上，一个 Java 应用程序的图形显示输出是一致的，其特征在于，它包括以下步骤:

在每个操作系统上，在 Java 应用程序中的一个给定执行点处，捕捉连续的图形显示输出帧;

为每个操作系统处理该连续图形输出帧，以生成一个Δ值，该Δ值为各操作系统屏蔽掉在该对帧上非时变的图形显示输出结构;

统计比较为各操作系统生成的Δ值，以确认在各操作系统上，Java 应用程序的图形显示输出是一致的.

10. 如权利要求 9 所述的方法，其特征在于，该 Java 应用程序是一个 Java 小程序。

11. 如权利要求 9 所述的方法，其特征在于，该处理步骤是一个按位异或逻辑操作。

12. 如权利要求 9 所述的方法，其特征在于，该比较步骤包括对Δ值执行一个统计分析。

13. 如权利要求 12 所述的方法，其特征在于，该统计分析是一个模式匹配算法。

14. 如权利要求 12 所述的方法，其特征在于，该统计分析是一个模糊逻辑算法。

15. 如权利要求 12 所述的方法，其特征在于，该统计分析是一个智能网络算法。

16. 一个计算机可读媒体上的计算机程序产品，用于确认一个给定软件应用程序的一个图形显示输出在各操作系统上是一致的，其特征在于，它包括:

在各操作系统上，在软件应用程序的一个给定执行点工作的装置，用于捕捉一对图形显示输出帧;

响应于该捕捉装置的装置，用于处理每对图形输出帧，以便为各操作系统生成给定信息，该给定信息为各操作系统屏蔽掉在该对帧上非时变的图形显示输出结构; 及

响应于处理装置的装置，用于比较为各操作系统生成的给定信息，以确定给定软件应用程序的图形显示输出在各操作系统上是一致的。

17. 如权利要求 16 所述的计算机程序产品，其特征在于，该给定软件应用程序是一个用 Java 编写的应用程序。

18. 如权利要求 16 所述的计算机程序产品，其特征在于，捕捉装置是一个图像帧抓取器。

19. 如权利要求 17 所述的计算机程序产品，其特征在于，该处理方式是一个异或逻辑操作。

20. 如权利要求 17 所述的计算机程序产品，其特征在于，该比较方法是一个统计分析。

21. 如权利要求 20 所述的计算机程序产品，其特征在于，该统计分析是一个模式匹配算法。

22. 如权利要求 20 所述的计算机程序产品，其特征在于，该统计分析是一个模糊逻辑算法.

23. 如权利要求 20 所述的计算机程序产品，其特征在于，该统计分析是一个智能网络算法.

24. 一个计算机，包括:

一个处理器，和

一个工具，用于确认一个给定软件应用程序的一个图形显示输出在各操作系统是一致的，它包括

在各操作系统上，在软件应用程序的一个给定执行点处工作的装置，用于捕捉一对图形显示输出帧;

响应于该捕捉装置的装置，用于处理每对图形输出帧，以便为各操作系统生成给定信息，该给定信息为各操作系统屏蔽掉在一对帧上非时变的图形显示输出结构，和

响应于处理装置的装置，用于比较为各操作系统生成的给定信息，以确认给定软件应用程序的图形显示输出在各操作系统上是一致的.

25. 如权利要求 24 所述的计算机，其特征在于，给定的软件应用程序是一个用 Java 编写的应用程序.

26. 如权利要求 24 所述的计算机，其特征在于，该捕捉装置是一个图象帧抓取器.

27. 如权利要求 24 所述的计算机，其特征在于，该处理方法是一个异或逻辑操作.

28. 如权利要求 24 所述的计算机，其特征在于，该比较方法是一个统计分析.

29. 一种方法，用于确认当一个当原来在一个面向目标、平台独立编程环境中写成的软件应用程序在运行第一和第二操作系统的各窗口环境中被执行时，该应用程序具有一个一致性的图形输出，该方法的特点在于:

在各操作系统的每一个上，在软件应用程序的一个给定执行点，捕捉一对图形显示输出帧;

对每个操作系统处理每对图形输出帧以生成给定信息，该给定信息为各操作系统屏蔽掉在该对帧中非时变的图形显示输出结构;

统计比较为各操作系统产生的给定信息，以确认该给定软件应用程

序的图形显示输出在各操作系统上是一致的.

30.如权利要求 29 所述的方法，其特征在于，该面向对象，独立于平台的编程环境是 Java.

31.一种方法，用来确认当一个软件应用程序在各测试系统配置中执行时，该程序的图形显示输出是一致的，其特征在于，它包括以下步骤:

当该软件应用程序在各测试系统配置上执行时，在软件应用程序的一个给定执行点，捕捉连续的图形显示输出帧;

为各测试系统配置，处理该连续的图形显示输出帧，以生成一个Δ值，该Δ值屏蔽掉在该对帧中非时变的图形显示输出结构;

将为每个测试系统配置所产生的Δ值进行统计比较，以确认该软件应用程序的图形显示输出在各测试系统配置中是一致的.

32.如权利要求 31 所述的方法，其特征在于，测试系统配置中至少有两个使用一个给定操作系统.

33.如权利要求 31 所述的方法，其特征在于测试系统配置中至少有两个使用不同的操作系统.

说　明　书

用于确认应用程序图形显示输出的方法和系统

本发明大体上关于计算机软件开发工具，具体地说，关于一种方法，用来确认一个特定软件应用程序的图形输出在多个不同类型的操作系统平台上是“一致的”.

Java，最早由 SUN 微系统公司开发，是一种面向对象的，多流处理的，可移植的，独立于平台的，安全的编程环境，用于开发、测试和维护软件程序. Jave 程序广泛用于万维网上，它是互联网的多媒体信息检索系统. 这些程序包括全特性人机对话，独立应用程序，以及被称为applet的较小的程序，它运行在一个能运行Java的网络浏览器或applet取景器中.

许多企业已开始写 Java 应用程序或 applets. 不过，这些公司在有不同操作系统的计算机系统中投资也很大. 这样，例如，一个特定的企业网可能支持一些运行不同操作系统的机器，如 IBM®、OS/2®、Microsoft Window95、Microsoft Window NT 4.0、UNIX、AIX、OS/400或其它类似系统. 这些基于 Java 的程序生成图形显示输出，这些输出包括一般的接口显示单元，如视窗、菜单、位图、图标和其它基本的控制，它们构成一个图形用户接口（ ‘GUI’ ），以及程序指定的显示单元. 总地或部分地说，这些单元为该应用定义了一个“图形输出”，且软件设计者的一个设计目的是，该应用软件的图形输出在不同操作系统平台（以及在有其它硬件/软件差别的同类平台）上是一致的. 目前，还没有有效的工具或方法，证实一个特定的基于 Java 的应用程序或 applet 的操作在多个操作系统平台上是相同的（或充分相似的）. 类似的，原有技术不能提供任何合适的技术，以确认一个 Java 应用程序或 applet 的多次执行的输出在一个给定的置信度范围内是“等效的”. 若没有这类工具，则很难使新的基于 Java 的应用程序和 applet 的测试自动化.

本发明用于解决该问题.

本发明的一个主要目的是确认一个给定软件应用程序的图形输出在多个操作系统平台上是一致的.

本发明的另一个主要目的是确认一个给定软件应用程序的图形输出在该应用在一个特定计算机配置中的多次执行间是一致的.

本发明的另一个主要目的是在多个操作系统平台上确认一个基于 Java 的应用程序或 applet，以简化一个企业计算机网中新的基于 Java 的应用程序的设计、开发、测试和实施.

本发明的另一个目的是便于用于开发新的软件应用程序的测试方法自动化.

本发明的另一个特定目的是，通过提供一个用于确认一个应用程序的图形显示输出的自动化工具，来增强一个 Java 程序开发环境.

本发明的另一个重要目的是，允许测试和确认 Java 独立于平台的代码.

本发明的一个副产品或优点是减少设计，开发和测试一个软件应用程序所必须的时间和努力，例如开发一个按独立平台编程环境写成的应用程序.

本发明的一个更一般的目的是减少与创建一个软件应用程序相关的开发时间和费用.

本发明的这些和其它目的在一个方法中得到，该方法用于证实一个给定软件应用程序的图形显示输出在各个操作系统上是一致的. 按该方法的一个最佳实施例，当一个软件应用程序在各个操作系统上执行时，捕捉在该软件应用程序的一个给定执行点上一组图形显示输出帧（例如，一对相邻帧）. 一个图像帧抓取器被用于该目的. 对各个操作系统，处理每对图像输出帧以生成一个差值或“Δ”值. 该特定Δ值包括这样的信息. 该信息为各操作系统屏蔽掉一些图形显示输出结构，当应用程序运行在给定执行点时，被屏蔽掉的图形显示输出结构在该对帧间是不改变，即非时变的. 这样的结构可以包括，例如，视窗框、标题条、滚动棒及其它这类显示控制单元. 按本方法，处理步骤包括在一个位－位的基础上，对帧执行一条逻辑操作（例如，一个“异或”）. 然后，比较各操作系统的Δ值以证实该应用程序的图形显示输出在各操作系统上是统一的或“一致的”. 如果一个给定的显示输出在一个给定置信度（例如 95％相似性）内是相似或匹配的，则该输出被称为是“一致的”. 最好，利用一个统计方法来执行Δ值的比较，例如一个模式配对算法，一个实施模糊逻辑算法的神经网络，或其它已知的或最近开发的统计算

法．另外，可以将一种或多种确定性方法用于该目的．

本发明的方法被用于确认任何类型的软件应用程序的图形显示输出，不过，一个特定的最佳实施例包括基于 Java 的应用程序或 applets 的测试和确认．

以上所述方法确认某应用程序的操作在不同操作系统平台上是相同（或充分相似）的．本发明技术也适用于确认一个给定应用程序（例如一个 Java 应用程序）在同类计算机系统（例如，一个有单一类型操作系统的计算机系统）上的多次执行的输出是等值的．本发明方法最好在某一点上与前面所讲的非常类似，即在应用程序“运行测试”期间，连续的显示输出帧在某特定“执行”点被捕捉．对于各次运行测试，处理连续图形输出帧以生成一个Δ值．该Δ值，如以上所提到的，屏蔽或“滤掉”每对帧间非时变的图形显示输出结构．利用一种统计方法，比较运行测试（或一组给定的运行测试）的Δ值以证实在多次运行测试组间，软件应用程序的图形显示输出是一致的．

以上已概述了本发明的一些更贴切的目的和特征．这些目的应被看作仅是举出本发明的一些更突出的特点和应用．通过以不同方式提供所述方法，或如将要描述的那样修改本发明，可以得到许多其它有用的结果．于是，参照以下对最佳实施例的具体描述，可以对本发明及其它目的有更全面的了解．

为了全面理解本发明及其中的优点，应参照以下与附图相联系的具体描述，附图中：

图 1 是一个有代表性的计算机的方块图；其中全部或部分地实施了本发明；

图 2 是本发明确认工具的基本操作的一个流程，用于证实在运行多种不同操作系统的计算机上的软件应用程序的图形显示输出；

图 3 是一个方块图，举例说明了本发明确认工具的主要处理功能；

图 3 是一个流程图，举例说明了图 2 中工具的最佳操作方法，且

图 4 是本发明确认工具的基本操作的流程图，用于在运行同类操作系统的计算机上的多次运行测试间确认软件应用程序的图形显示输出．

如在以下将要描述的，在一个计算机或通过计算机网络互连的一个或多个计算机上，可以执行构成本发明“确认”工具的一个或多个过程．参照图 1，示出了用于实施本发明的一个计算机．该计算机 10 有一个处

理器 12，一个操作系统 14，一个操作系统应用程序接口（API）16，一个图形用户接口 18 和 RAM 22. 在本发明的一个实施例中，本工具用于确认基于 Java 的应用程序和/或 applet，这样，可以选择地，该计算机还包括一个 Java 虚拟机（JVM）解释器 20. JVM 是一个抽象的计算机，它包括一个指令集并使用不同的内存区域. 在许多现有的互联网浏览器应用程序中，例如 Netscape Navigator™和 Microsoft® Internet Explorer™的合适版本中，可以得到一个 JVM. 关于 JVM 的进一步的细节见于 Java™虚拟机指南，Tim Lindholm 和 Frank yellin，Addison Wesley(1997), ISBNO-201-63452-X，在此引入以供参考.

这样，用于本发明的计算机可以是任何基于 Intel®, Power PC®或 RISC 的个人计算机或工作站用户，或是服务平台，它包括一个操作系统，例如 IBM®OS/2®，Microsoft Windows 95，Microsoft Windows NT4.0，Unix，AIX ®，OS/400 或类似的操作系统. 一个有代表性的计算机运行一个 Intel X86 处理器，OS/2 Warp 4.0 版操作系统，及 JVM 1.0.2 版或更高版. 换一种方式，该计算机运行基于 X86 的处理器，Windows '95（或 Windows NT）操作系统，及 JVM 1.0.2 版或更高版. 另一种替换方式是运行一个有 Power PC 或基于 RISC 处理器的计算机，AIX 操作系统，及 JVM 1.0.2 或更高版.

确认工具 25 最好以软件实施. 更具体地，以 Java 写成以便由 Java 虚拟机解释器 20 执行. 尽管确认工具 25 最好是基于 Java 的，但这并不是本发明的一个要求. 不过，基于 Java 的确认工具在多个开发平台上是可移植的. 这样，确认工具码本身在每个以上所述系统中是可操作的，尽管这些系统至少在操作系统级上是异构的. 确认工具最好具有一个与此相关并由计算机控制的测试子系统 26. 测试子系统 26 一般包括合适的硬件，例如一个或多个图象帧“抓取器”28，用于在计算机用户接口上捕捉图象帧输出，以及任何必须的接口，控制和同步电路及部件. 图象帧抓取器的技术是已知的. 测试子系统 26 的一些或所有部件和/或功能，都可以全部或部分地以软件实施，或通过使用计算机自己的内部资源及外围设备来实施.

确认工具 25 一般用于确认正在开发（或正在测试）的一个给定应用程序的操作在多个操作系统平台上是否有一个一致性的显示输出；另一方面，确认工具适用于确认应用软件的图形显示输出在有同一操作系

统的一个系统上的多次执行中是否一致．如本技术中所熟知的，一个与常规的“基于视窗”图形用户接口相关的显示输出包括一些单元，而这些单元非限制地包括窗口，菜单，标题棒，框，滚动条，位图，图标及构成 GUI 的其它控制．在独立于平台的软件程序（例如 Java 应用程序或小程序）的开发和设计中，开发者（或其他人）希望能够确认正在开发的特定软件在多种不同计算机系统中是否呈现同样的“外观和感觉”．确认工具 25 提供了一个有效和高度精确的机制，以确认一个独立于平台的应用程序在已知图形用户接口上呈现一个一致性的图形显示输出，已知的图形用户接口例如，IBM OS/2 图象管理器，Microsoft Windows NT,Microsoft Windows 95，Macintosh，和 X Windows，以及其它接口．

图 2 是本发明确认工具的基本操作的流程，在运行不同操作系统的计算机上确认软件应用程序图形显示输出．为讨论起见，以下所做讨论的范围是只在两个操作系统上测试应用程序．不过，对于一个具有一般技术的人来说，可以认识到这并非是本发明的限制，因为多次运行测试可以包括任意数量的异构计算机平台．

基本的例程包括一些处理步骤，它们可由计算机中的硬件或测试子系统执行，或在确认工具软件自身中执行．在步骤 30 开始例程，确定是否所有的运行测试都已结束．如以上所提到的，一个给定的运行测试包括一些（将在以下介绍），对在测试程序所测试的一个给定操作系统平台上运行的一个给定应用程序的处理步骤．如果在第 30 步，该测试的结果是肯定的，则例程分支到第 40 步，该步将在以下做详细介绍．如果在第 30 步，该测试的结果是否定的，则进程在第 32 步继续，运行下一次测试．以下步骤描述了基本操作．

在第 34 步进行一次检测以确定正在被测试的应用程序是否已到达一个给定执行点．按本发明，一个由确认工具测试的应用程序是由多次运行测试来检测的，但都在一个给定的执行点．以这种方式，开发者可以确信，在给定执行点不改变的指定操作 GUI 结构可以被从分析中滤出（最好，如将要看到的那样，使用一个统计分析）．给定执行点最好可以由用户提供，或是可以由一些系统控制改变．如果在第 34 步，检测的结果是否定的，则例程循环并继续重复该步骤．不过，如果在第 34 步检测的结果是肯定的，它表明特定的运行测试已到达给定执行点，然

后，例程在第 36 步继续.

此时，例程处理被测试应用程序生成的一对图形输出帧. 在一个最佳实施例中，利用测试子系统中的图象帧抓取器捕获该对帧，这对帧代表正好在给定执行点（应用程序中）之前显示在 GUI 上的特定的帧和正好在给定执行点之后显示在 GUI 上的特定的帧. 一个帧可以包含显示图象的所有或任何给定部分. 所选的特定帧不需是连续帧或是具有规定的特定时间关系，不过，其条件是，在多次运行测试中要使用同一帧配对关系.

然后，例程在第 38 步继续，处理这对图形输出帧. 按照一个最佳实施例，第 38 步包括对该帧做按位异或，以便为特定运行测试生成一个Δ值. 这样，将为该次特定运行测试生成Δ值（这里，该应用程序是在一个第一给定操作系统上执行的）. 尽管最好对整个帧做按位异或逻辑操作，但这并不是本发明的要求. 这样，例如，可以以这种方式处理帧对的任何给定部分，只要随后在其它运行测试中检测同一给定部分. 另一方面，可以使用一些其它逻辑操作，而非异或，只要在其它运行测试中也实施类似操作.

以这种方式处理这对帧具有显著优点. 特别地，在帧对间不改变的所有 GUI 显示结构基本上被屏蔽掉或滤去，只留下存在于特定执行点上的帧的“差值”. 这样，包括视窗框，标题棒，滚动棒，图标等，以及其它系统指定操作的输出信息的这类显示结构被从确认处理进一步的考虑中消去.

在第 39 步保存Δ值以后，控制回到第 40 步. 在该步，Δ值 {Δ1, Δ2, ……} 被从存储器中取出. 这些值的每一个都代表连续的帧对输出（即，从帧“n-1”到帧“n”）在应用程序中同一给定执行点处不同的程度（当该应用程序在多个操作系统平台中指定的一个上执行时）. 在第 42 步，例程挑选一个给定的统计方法来分析该Δ值. 按本发明，可以使用任何合适的统计方法来比较在多次运行测试中生成的Δ值. 已知的统计方法包括，不加限制，模式匹配算法，模糊逻辑算法，一个具有一组学习规则的自适应推理机，及其它已知或最近开发的统计估计例程. 在运行在计算机上的软件中实施该特定例程. 如果希望，该步骤可以包括任何已知或最近开发的确定性分析方法.

在第 44 步，对该组的Δ值应用统计方法. 然后，在第 46 步运行一

个测试以确定在一个给定的，用户可选的置信度（例如 95％）内，Δ值是否“匹配”．当然，可以使用任何种类的统计量度．如果在第 46 步，检测的结果是否定的，则在第 48 步向用户提供一条指示，表明在被测试的多个操作系统平台上，该应用程序的图形显示输出是不一致的．如果希望，可以在第 50 步执行一个附加步骤，以隔离并标识特定的显示输出的差异．不过，如果在第 46 步，检测的结果是肯定的，则在第 52 步，例程继续，生成一个指示，表示图形显示输出在多个被测试操作系统中是“正确的”，例程结束．

这样，如在图 3 的方块图中所举例说明的，每次运行测试 54 生成一个Δ值Δn．然后，不同的Δ值被提供给统计分析器 56 以生成一个“匹配”或一个“舍弃”．另一个替换输出可以是，结果是“非确定的”，这就需要重复一次或多次运行测试．该工具还可以包括一个诊断例程，以估计并输出一些特定信息，该信息表明哪些特定图形显示输出信息影响或引起拒收．该信息可被开发者用于纠正问题（例如，通过再加工应用程序代码）．

这样，当一个测试中的应用程序在多个不同的操作系统平台上运行时，本发明便利地捕捉连续的图形显示输出帧．然后，该帧被处理以屏蔽掉所有在给定执行点非时变的操作系统指定 GUI 及程序生成结构．通过利用一个给定的统计分析，比较为每次运行测试生成的Δ值，确认工具证实程序输出的一致性．这显著缩短了程序开发时间，精力和费用，特别是对于独立于平台的应用程序．要利用该工具测试和调试的一个特定应用程序是一个 Java 应用程序或小程序．在是一个 Java 小程序的情况下，一个小程序观测器可被用于进一步证实一致性．

图 4 举例说明了本发明一个改进方案的流程图，其中，在一个同类平台上，在连续的运行测试中测试一个给定应用程序．该实施例使应用程序开发者能够证实，该应用程序在利用相同或基本类似的执行环境的多次执行中，操作是一致的．这样，在图 4 的实施例中，希望操作平台是相同的．

如前面的实施例那样，基本的例程包括由计算机或测试子系统 26 中的硬件，或由确认工具软件自身执行的处理步骤．在第 60 步，例程开始，确定是否所有的运行测试都已完成．在该实施例中，一个给定的运行测试包括一些处理步骤（以下将要描述），这些步骤有关一个在同

一操作系统平台（但可能有不同的硬件配置）上执行的给定应用程序．如果在第 60 步，检测的结果是肯定的，则例程分支到第 70 步，该步骤将在以下详细介绍．不过，如果在第 60 步，检测的结果是否定的，则例程继续进行到第 62 步，运行下一次测试．

在第 64 步，进行检测以确定正在被测试的应用程序是否已到达执行点．如果在第 64 步检测的结果是否定的，则例程循环并继续重复该步骤．不过，如果在第 64 步检测的结果是肯定的，它表示特定的运行测试已到达给定执行点，则例程继续进行到第 66 步．

正如前面根据图 3 所示例程描述的那样，例程在第 68 步继续，处理被图象帧抓取器捕获的图象输出帧对．按最佳实施例，第 68 步又包括对帧进行按位异或，从而为特定的运行测试生成一个Δ值．这样，就为这一次特定的运行测试生成了Δ值Δ1．然后，在第 69 步保存该Δ值，且控制返回第 70 步．在这一步，从存储中取出Δ值{Δ1，Δ2……}．在第 72 步，例程选择一种给定的统计方法以分析该Δ值．

在第 74 步，对该组Δ值应用统计方法，然后，在第 76 步运行一次检测以确定在一个给定的，用户可选的置信度内，该Δ值是否“匹配”．如果在第 76 步，检测的结果是否定的，则在第 78 步向用户提供一个指示，告诉用户该应用程序的图形显示输出在被测试的系统平台或配置上是不一致的．如果希望，在第 80 步执行一个附加的步骤，以隔离并标识特定的显示输出差值．不过，如果在第 76 步检测的结果是肯定的，则在第 82 步，例程继续，生成一个指示，告诉用户该图形显示输出在被测试的多个操作系统上是“正确”的．如果检测的结果不确定，则例程也提供一个合适的指示，这使得处理过程结束．

对本技术有一般知识的人士，可以认识到，可以对本发明改动或修改，且仍在所述范围内．例如，有可能希望确定各对帧（在一个给定的运行测试中）间相似的程度，而不是让确认工具生成一个“差值”或Δ值．这样，当希望用“差值”技术时，它并非是对本发明的限制．进一步，尽管对Δ值或其它值做统计分析是确认测试结果的优选方法，但有一般技术的人士都可以认识到，也可以使用确定性分析技术．所有这些改变都是在本发明范围内的．

本发明的一个最佳实施例是一个应用程序，即一组代码模式的指令（程序代码），它可以，例如，位于计算机的随机存取存储器中．该组

指令被保存在另一个计算机存储器中，直到被计算机请求，例如，保存在一个硬盘设备中，或在一个可移开存储器中，例如一个光盘（最终用于一个 CD ROM）或软磁盘（最终用于一个软磁盘驱动机构），或是通过互联网或其它计算机网络下载．这样，本发明可被实现为用于计算机中的一个计算机程序产品．

另外，尽管所述的不同方法是简单地实现在一个通用计算机中，由软件有选择地启动或重新配置的，但对本技术有一般知识的人士能够认识到，这种方法可以用硬件，用固件，或用被构造为去执行所需方法步骤的更专用的设备来执行．

尽管没有要求，但组成本发明的不同进程应位于同一主机上或位于由一个网络（例如，Internet，Intranet，广域网（WAN）或局域网（LAN））互联的不同机器上．这样，一个运行本发明的机器具有合适的网络硬件，以建立与其它计算机的连接．例如，一个计算机可能有一个到运行在一个令牌环或以太网适配器上的网络的 TCP/IP 或 NETBIOS 连接．

那些本技术专业人士可以认识到，以上所述的指定实施例可以很容易地被用做修改或设计用于完成与本发明目的相同的其它技术的基础，还可以认识到，这类等效结构并不脱离附加权利要求中所陈述的本发明的精神和范围．

这里阐述了本人的发明，本人所声明为新的和希望由特许证书保护的内容在以下权利要求中陈述．

说 明 书 附 图

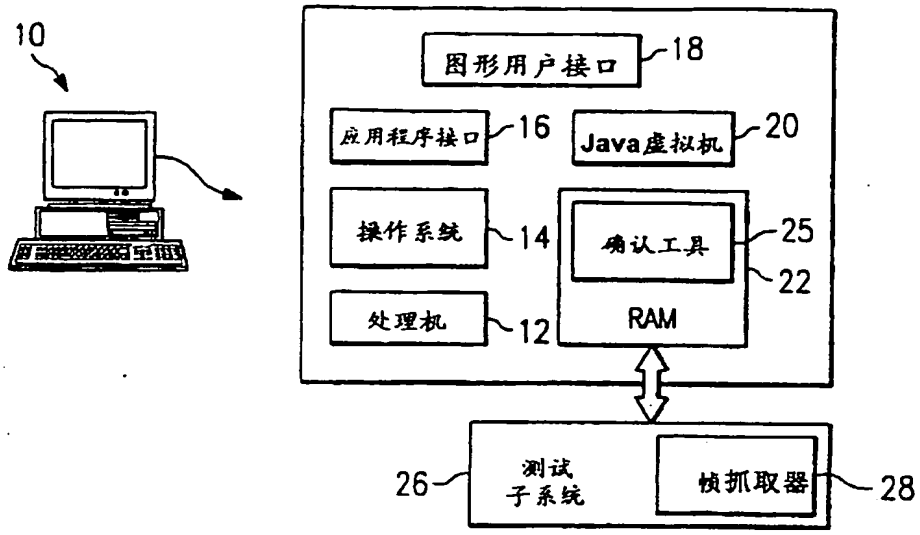

10
图形用户接口
18
应用程序接口
16
Java虚拟机
20
操作系统
14
确认工具
25
22
处理机
12
RAM
测试
子系统
26
帧抓取器
28

图 1

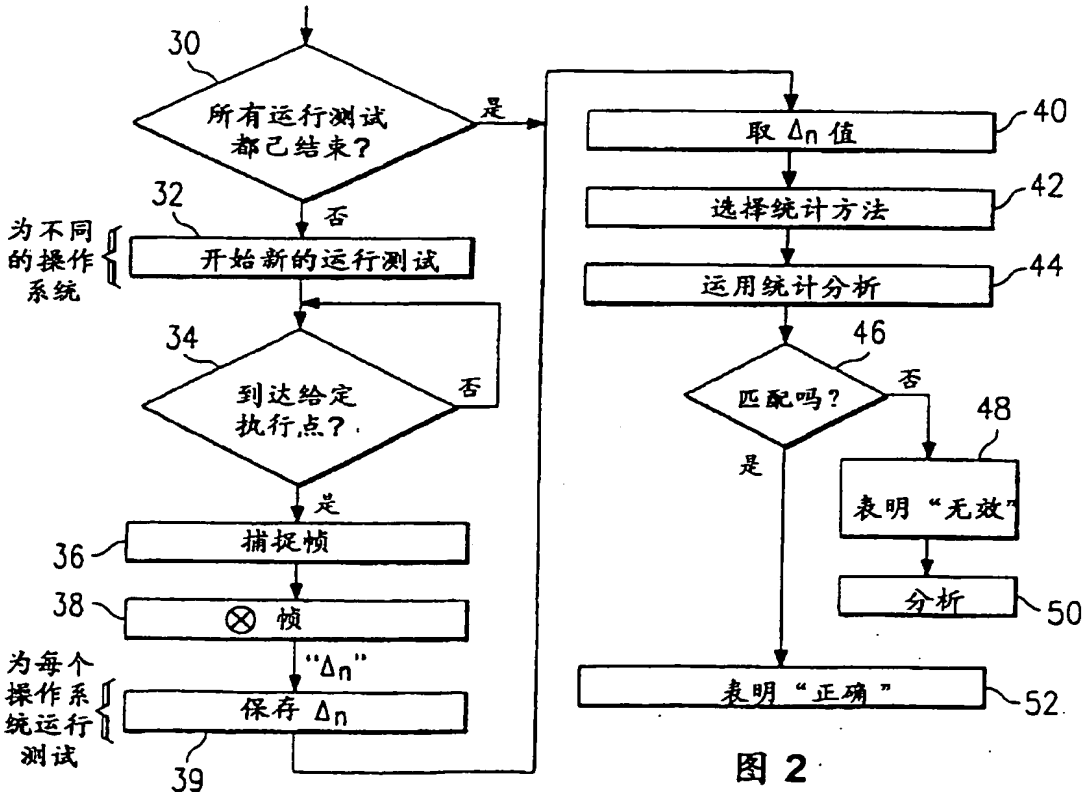

30
所有运行测试
都已结束?
是
否
32
为不同
的操作
系统
开始新的运行测试
34
到达给定
执行点?
否
是
36
捕捉帧
38
⊗ 帧
"Δn"
为每个
操作系
统运行
测试
保存 Δn
39
取 Δn 值
40
选择统计方法
42
运用统计分析
44
46
匹配吗?
否
48
是
表明"无效"
分析
50
表明"正确"
52

图 2

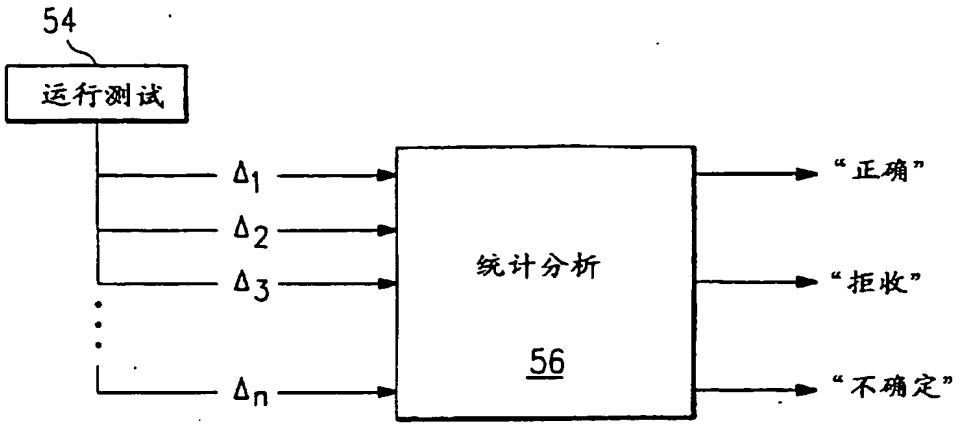

54
运行测试
Δ1
Δ2
Δ3
Δn
统计分析
56
“正确”
“拒收”
“不确定”

图 3

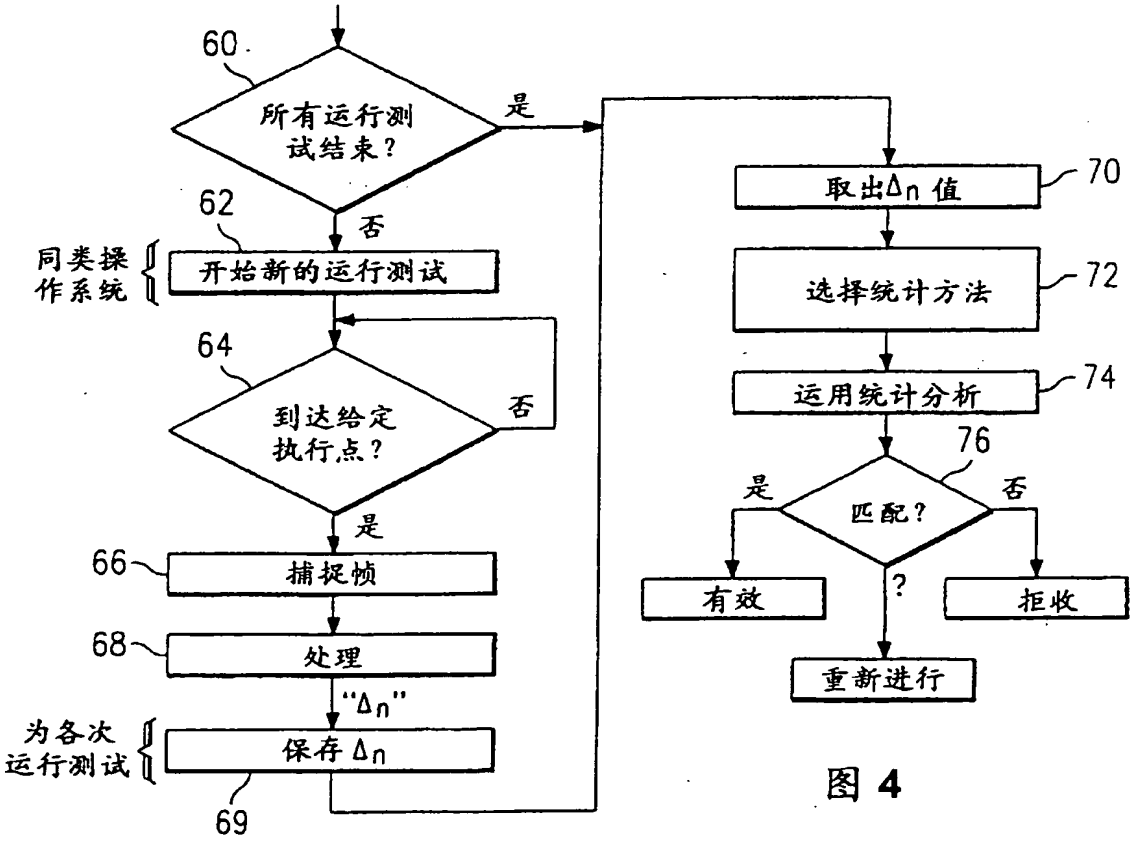

60
所有运行测试结束？
是
否
62
同类操作系统
开始新的运行测试
64
到达给定执行点？
否
是
66
捕捉帧
68
处理
“Δn”
为各次运行测试
保存 Δn
69
取出Δn 值
70
选择统计方法
72
运用统计分析
74
76
匹配？
是
否
有效
？
拒收
重新进行

图 4